ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Project: Design, Build and Control of a Jumping Robot

Team #36

XINYI YANG
(xinyiy19@illinois.edu)
SIYING YU
(siyingy3@illinois.edu)
HANJUN LUO
(hanjunl2@illinois.edu)
XUECHENG LIU
(xl125@illinois.edu)

TA: Leixin Chang

May 29, 2025

Abstract

Complex and uneven terrain presents significant challenges to the locomotion systems of traditional robots. Some researchers have proposed equipping robots with jumping mechanisms to address this issue; however, existing jumping robots often suffer from drawbacks such as high cost, difficulty in miniaturization, structural complexity, and maintenance challenges. To address these limitations, this project introduces a novel, low-cost, and reliable bio-inspired jumping robot designed to overcome these issues by mimicking the jumping mechanism of fleas. Our system integrates an efficient spring-based energy storage module, an intelligent actuation and control mechanism incorporating a computer vision module and motors, and an innovative trigger-release system, ensuring powerful yet precisely controllable jumps within a compact form factor. The integrated system, which includes mechanical, control, and computer vision modules, aims to achieve multi-level jumping capabilities, precise jump force regulation guided by real-time visual feedback, and accurate 3D environmental perception for adaptive trajectory planning. This research aims to provide a versatile robotic platform with enhanced agility and maneuverability for operation in challenging real-world application environments.

Contents

1 Introduction

1.1 Background and Objective

In real-world applications where robots are expected to operate beyond flat laboratory floors—such as in cluttered environments, uneven terrain, or areas with vertical obstacles—conventional wheeled or legged platforms often suffer from poor mobility and adaptability. This motivates the development of jumping robots, which can offer compactness, rapid vertical displacement, and terrain-independent locomotion [1, 2].

Drawing inspiration from nature, particularly from insects such as fleas which utilize specialized elastic energy storage structures to generate jump forces exceeding 100 times their body weight, our project aims to design and implement a bio-inspired, spring-powered jumping robot capable of achieving controlled, variable jump heights. We will explore the feasibility of using mechanical energy for impulsive vertical motion by emulating these explosive elastic energy release mechanisms. Our proposed robot integrates a torsional spring-based energy storage mechanism, a gear-driven or cam-based trigger-release system, and a lightweight structural chassis.

1.2 Functionality

The project is designed to meet the following high-level requirements. Each requirement is carefully chosen to align with our core design purpose: creating a compact, agile, and controllable jumping robot suitable for complex environments.

- **Compact Dimensions:** The overall robot dimensions must not exceed $15 \times 15 \times 15\,\mathrm{cm}^3$.
  This size constraint directly addresses the objective of compactness, a primary advantage of jumping robots highlighted in our purpose. A small footprint is essential for navigating cluttered environments and constrained spaces where conventional robots fail, reflecting the core motivation of our project. It also aligns with the bio-inspiration from small, agile organisms.
- **Controlled Jump Height:** The system must support two levels of jump height, with at least one exceeding 10 cm, and maximum height over 30 cm.
  Achieving a significant jump (exceeding 10 cm) demonstrates the capability for rapid vertical displacement, allowing the robot to overcome vertical obstacles – a key performance goal. More importantly, the requirement for controlled, variable jump heights enhances both adaptability and terrain-independent locomotion, enabling the robot to tailor its movement to specific challenges.
- **Wireless Triggering:** The jump must be triggered wirelessly via Bluetooth.
  Wireless triggering is fundamental for untethered operation in realistic, complex scenarios. It enables remote activation, crucial for deployment in inaccessible areas and for achieving rapid response to sudden mobility demands. This autonomy removes physical constraints, maximizing the robot's adaptability and utility beyond controlled laboratory settings.

1.3 Subsystem Overview

The system is divided into three main subsystems: the CV (Computer Vision and Communication), Control, and Motion modules. The CV module allows the user to send a jump command via Bluetooth. The control module, based on an ESP32 microcontroller, receives the command and drives a servo motor to preload the spring mechanism. Upon reaching the trigger angle, a half-gear mechanism releases the stored energy to initiate the jump. The motion module consists of a four-bar linkage, gear transmission, and two parallel torsional springs responsible for energy storage and mechanical release.

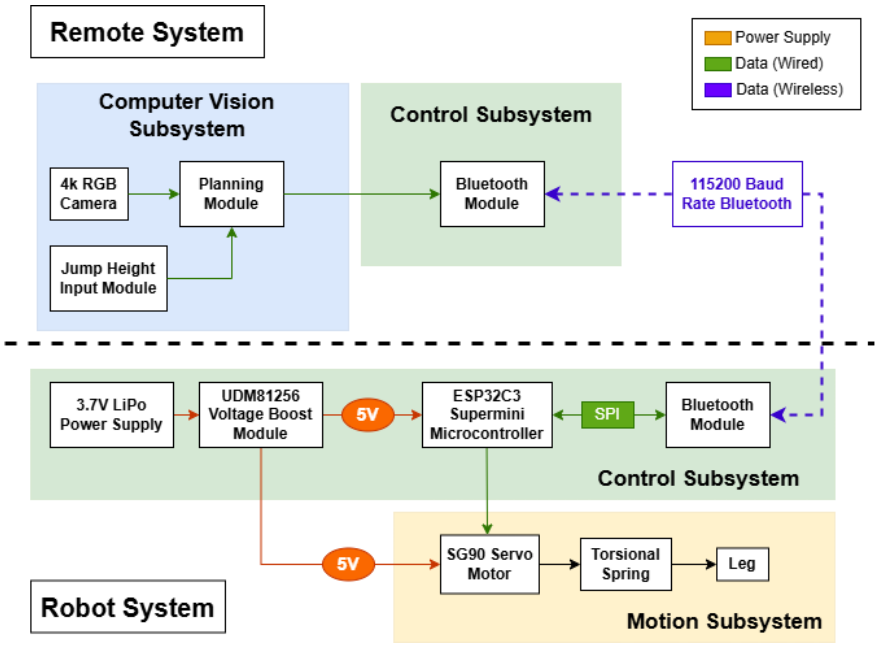


Figure 1: Block Diagram of our jumping robot.

The block diagram Figure 1 decomposes the platform into three interacting subsystems that are separated by a dashed line: everything above the line resides on the remote computer while the hardware below forms the on-board robot.

2 Design

2.1 Design Procedure and Alternatives

2.1.1 Mechenical Design

The robot's mechanical system evolved through three main phases, moving from initial concepts to a refined final design. The core design consistently centered on a four-bar linkage for jump actuation and torsional springs for compact energy storage.

Phase I: Initial Concept and Prototype Development In this initial phase, the foundational four-bar linkage was designed to maximize vertical displacement, and torsional springs were selected for their efficient form factor. Early prototypes, however, revealed critical issues: an initial release mechanism using an eccentric wheel and a dedicated motor proved unstable and heavy; plastic shafts used for connections were too weak for the spring torque; and a 3D-printed Y-shaped bracket for base support was too wide and heavy, limiting jump height. Complex solutions for dynamic leg length adjustment, such as a worm gear, were also considered but assessed as too heavy and complex.

Phase II: Addressing Critical Failures and System Simplification This phase focused on resolving the most pressing issues. The problematic eccentric wheel release system was abandoned, and development began on a custom half-gear structure, a passive design aimed at reducing weight and complexity while improving release reliability. Recognizing the failure of plastic shafts, the decision was made to upgrade to metal shafts. Furthermore, the strategy for leg length adjustment was simplified from dynamic concepts to manual sliding slots, allowing for pre-jump tuning without the extra weight of motor. The base support, however, still awaited a more optimal solution.

Phase III: Performance Optimization and Final Design The final phase implemented all necessary upgrades and validated overall performance. The custom half-gear passive detachment system was fully implemented and refined, proving to be a simple, lightweight, effective, and stable solution for energy release. Metal shafts were integrated, providing the necessary structural integrity against high torsional forces. A critical improvement was the replacement of the 3D-printed Y-bracket with lightweight, strong carbon fiber rods for base support; this significantly improved stability and dramatically increased the maximum achievable jump height from approximately 10 cm to 40 cm. The manual sliding slots for leg length adjustment were also finalized, offering a practical means to configure varied jump performance. This iterative process resulted in a robust mechanical design that achieved targeted performance enhancements.

2.1.2 Embedded Design

Phase I: Stepper and Coreless Motor Prototype The first prototype was a two-motor design, utilizing a stepper motor to actuate the release and using a coreless brushed DC motor to compress the spring. The stepper motor used full-step drive mode, allowing

for accurate position control and triggering the release part of the constraint. The coreless motor did not produce enough torque to compress the spring through the gear train, even when duty cycles were at the maximum. As a result, the cumulative weight and size of the two motors and supporting control electronics exceeded our limits for a lightweight mobile platform. In addition, in order to quickly test and iterate, we added an infrared control module to this generation of control system.

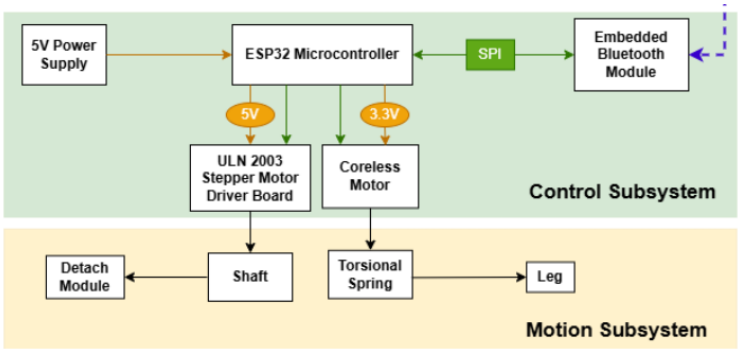


Figure 2: Block Diagram for Two Motor System

Phase II: Servo Motor and ESP32 Development Board To simplify the mechanical and control structures, the second iteration replaced the two motors with one continuous 360° SG90 servo. The servo offered bi-directional continuous rotation with ample torque to compress and release the spring and reduced the mechanical complexity significantly. To generate PWM signals and facilitate bluetooth communication with a host PC, an ESP32-WROOM development board was used. While this configuration worked properly, the development board's large form factor and peripheral overhead were not acceptable for onboard use in the compact robot chassis.

Phase III: Servo Motor and ESP32-C3 SuperMini Integration We used the ESP32-C3 SuperMini, a small and lightweight microcontroller module, to control the SG90 servo. This allowed us full integration of the embedded system onto the robot, while still maintaining full functionality of the BLE and PWM generation. The ESP32-C3 SuperMini was powered by a HW-085 boost converter that stepped up the 3.7V output from a single cell (401119-100) LiPo battery to a steady 5V. This 5V rail powered both the servo and the ESP32-C3 board. The total weight of this system is about 5g. On the prototype, we connected the ESP32C3, boost module and servo by soldering, and connected the battery and boost module by terminals. After testing, the prototype can run successfully.

For a more compact integration, we also designed a custom PCB, as shown in fig 3 to consolidate the boost converter, servo headers, battery connector, and ESP32-C3 interface. The EN pin of the boost converter was connected to a toggle switch so that power could be turned on and off conveniently. The size of the PCB is 2.6cm * 3cm, which is very small and can be easily integrated into the prototype.

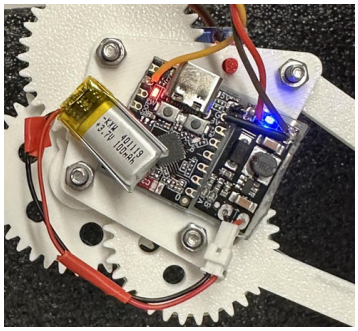
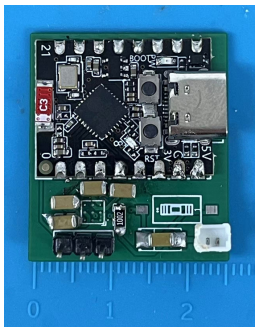



Figure 3: Soldered Boards and Customized PCB for Phase III

2.1.3 Computer Vision Design

The computer vision module of our jumping robot comprises a UHD RGB camera, a computing unit, and associated algorithms. Our selection choices were guided by the various limitations associated with our small-scale jumping robot, real-time capability, cost, and dependent variable in the controlled test situation. The development went through several stages in the process to achieve the current design.

Phase I: Camera Type The initial phase focused on selecting an appropriate method for 3D distance detection. We evaluated several approaches, including stereo camera systems, LiDAR, and monocular vision with fiducial markers. Ultimately, we selected a single high-resolution RGB camera in conjunction with Aruco markers [3]. Stereo systems give you precise depth perception directly, which resulted in richer 3D as well, but they are typically more complicated to calibrate and have a higher compute cost. While LiDAR can measure accurate distances, it also have higher prices and potential for larger form factors. Aruco markers gave precise and robust 3D distance information by using a single standard RGB camera. This seemed like a good trade-off and also gave us enough accuracy for the simple navigation and jump control tasks we would perform in a controlled-environment [4]. The high resolution of the UHD camera was chosen to ensure reliable detection of the Aruco markers, even at moderate distances.

Phase II: Camera Position During the second phase we discussed about the physical positioning of the camera. An important consideration was the balance of a fully on-board vision system versus off-board. Initially, we considered miniaturizing the camera and the processing unit and mounting them on the robot, but we ran into several critical problems. First it was obvious that our robot is very light to help maximize jump performance. Adding the weight of an on-board camera, a processing unit for real-time video analysis, or a wireless transmitter with enough bandwidth to stream the raw video, had the potential to significantly restrict the robot's achieved agility and height. Second, to come into play of superimposing such a high performing vision system took a availability of a mechanical design that was so complex and high power and ultimately had effects on the cost of the robot, its power supply system, and thermal constituents. Therefore, we took a decision to use an external system. This meant a stationary high resolution RGB

camera connected to a high performance laptop that performs all of the processing. This still keep the robot being lightweight. Another benefit of this configuration is that it does not increase the robot's on-board electronics or power requirements.

Phase III: Distance Format The last design phase was about the representation of target distance and position. We explored a polar coordinate system by 2D image cues, angle and direct line-of-sight distance, which was appealing for its simple conceptually and computational simplicity. This proved to be inadequate for defining the target distance and position for our 3D maneuvering robot due to built-in ambiguities for defining a 3D point with a 2D projection. Therefore, we used a complete 3-dimensional Cartesian coordinate system to define the target position relating to a well-defined point-of-origin. The 3D representations are a spatial description that may be uniquely mapped to the jump models and therefore allowed us to convert our height, distance, and orientation, into jump space and utilize unambiguous calculations of jump vectors.

2.2 Design Details

The overall layout and physical structure of the jumping robot are depicted in Figure 8. This section will provide a detailed introduction to the key components: the Computer Vision subsystem, the Control subsystem, and the Motion subsystem. These subsystems collectively form the complete operational pipeline for the robot, encompassing the entire process from environmental perception (Computer Vision), through decision-making and command generation (Control), to the physical execution of the jump (Motion).

2.2.1 Computer Vision Subsystem

The Computer Vision Subsystem is responsible for providing real-time spatial awareness, enabling the robot to perceive its environment, determine its pose, and identify targets or obstacles. It comprises a high-resolution UHD RGB camera, a high-performance computing unit, and a specialized dynamics model and corresponding algorithm. The workflow of our module is shown in Figure 4.

Imaging Sensor (Camera) A UHD (3840x2160 resolution) RGB camera operating at 30 frames per second (Hz) serves as the imaging sensor. This camera provides detailed visual information necessary for detecting small features and markers from a distance. The high resolution and adequate frame rate are crucial for capturing clear images of the operational environment. We employ OpenCV to interface with this camera in real-time, capturing images for subsequent processing. Due to the space limit, we provide detailed formulations for different scenarios in Appendix C.

Camera Calibration To ensure accurate 3D distance detection, a rigorous camera calibration procedure is performed prior to deployment. We use a standard 5×7 checkerboard pattern, as exemplified in Figure 5. 40 images of this checkerboard are captured from various angles and distances relative to the camera. OpenCV library functions are

then utilized to calculate the precise intrinsic camera parameters from these images. The calibration process solves for the camera's intrinsic matrix K and a set of distortion coefficients D. These parameters are subsequently used to undistort the captured images and are fundamental for accurately projecting 3D world points to 2D image coordinates and vice versa, which is essential for 3D reconstruction tasks.

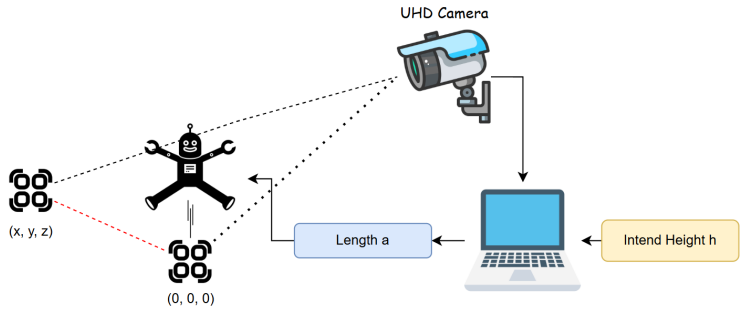


Figure 4: Workflow of our computer vision module.

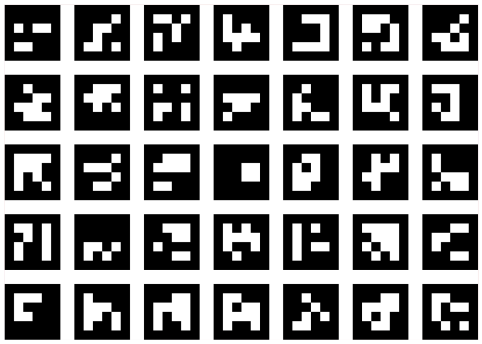


Figure 5: Our calibration board.

Aruco-based Distance Detection For robust and efficient tracking of the robot and environmental features, we employ 4×4 Aruco markers from the `DICT_4X4_250` dictionary provided by OpenCV [5]. These markers, with a physical size of 3cm×3cm, are strategically placed on the robot's body and at key locations within the test environment. The combination of the UHD camera and these distinct markers enables reliable detection at distances up to 3 meters. This marker system was chosen for its optimal balance between detection reliability, computational efficiency, and ease of implementation.

Image Processing and Computation The software processing utilizes OpenCV's ArUco module (version 4.5.0) to detect and decode markers. Adaptive thresholding techniques are employed within this module to robustly handle varying lighting conditions while maintaining a false positive detection rate below 0.1%. All computations are performed on a high-performance laptop equipped with an Intel Ultra 9 275HX CPU and an Nvidia RTX 5080 Laptop GPU, serving as the dedicated computing unit. To meet the real-time processing demands, specific optimizations were implemented by utilizing PyTorch and OpenCV libraries that are compatible with CUDA [6], thereby leveraging GPU acceleration for demanding tasks. This optimized pipeline successfully reduces the processing latency for each 4K resolution image to less than 100ms. The 3D distance D to a detected Aruco marker is then calculated using the pinhole camera model principles:

$$D = \frac{W_{real} \times f_{pixel}}{W_{pixel\_image}}$$

where W_{real} is the known physical width of the Aruco marker (e.g., 0.03m for the 3cm×3cm markers), f_{pixel} is the camera's focal length in pixels (obtained from calibration), and W_{pixel_image} is the marker's apparent width in the image in pixels.

Jump Parameter Calculation Based on the acquired 3D distance information, we calculate the optimal configuration for the robot's variable four-bar linkage to achieve the desired trajectory. The primary output is the length 'a' of the variable link (specifically, the bottom-most bar of the linkage), which must be set within a predefined operational range. This calculation relies on known robot parameters: the kinetic energy E_k available at launch (assumed constant per jump), the mass m of the robot (and thus its weight $W = mg$), and the lengths of the three fixed links of the four-bar mechanism (l_1, l_2, l_3). The initial launch velocity magnitude v_0 is determined from the kinetic energy:

$$v_0 = \sqrt{\frac{2E_k}{m}}$$

This v_0 forms the basis for subsequent trajectory calculations, assuming negligible air resistance. The relationship between the variable link length a and the resulting launch angle θ_{launch}, denoted as $\theta_{launch} = f_{linkage}(a, l_1, l_2, l_3)$, is crucial and is derived from a detailed kinematic model of the robot's specific four-bar linkage.

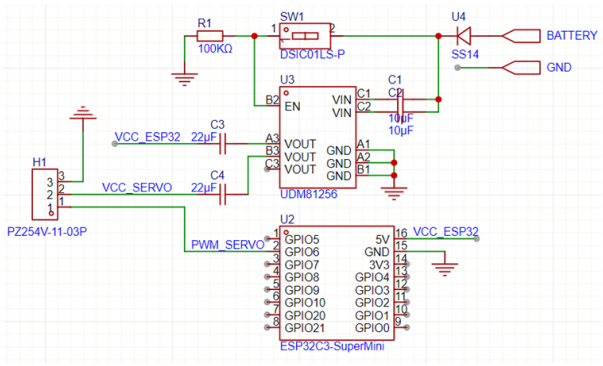
2.2.2 Control Subsystem

The Embedded Control System coordinates communications and motion for the robot. We used an ESP32-C3 development board soldered onto a custom PCB which is powered by a small 401119 LiPo battery. The commands are sent to the ESP32-C3 microcontroller wirelessly over Bluetooth Low Energy (BLE) from the host device. The microcontroller processes the received commands and generates PWM signals to drive the servo motor. All connection routing and power management is handled on the custom PCB.

Microcontroller The microcontroller, ESP32-C3 Supermini development board, handles data transmission and motor driving. This microcontroller was chosen because of its low cost, small size, light weight, and built-in Bluetooth capability. It communicates with the developer's computer and flashes the firmware through the type-C serial port.

To facilitate compact installation and autonomous operation, we designed a custom PCB, as shown in Figure 6, to connect the ESP32-C3 SuperMini to peripheral components. The PCB incorporates a boost converter module, a toggle switch, and connections to the servo, while providing a clean way to route power and deliver signals. It provides 3.7V regulated voltage from the 401119 LiPo battery to the servo through the boost module, while also consolidating all control and power paths into a single lightweight board.

Bluetooth The Bluetooth communication channel serves as the wireless bridge between the host PC and the embedded control system. This module is logically divided into two parts: the embedded device side, implemented on the ESP32 microcontroller, and the host-side controller, which initiates and manages the communication session. Together, these subsystems form a low-latency control loop that supports command dispatch and feedback monitoring, as shown in Figure 7. The embedded Bluetooth Low Energy (BLE) module is implemented on the ESP32-C3 using the native BLE stack. The ESP32 acts as



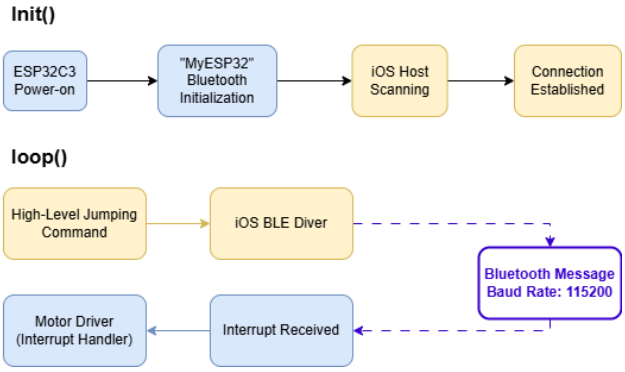


Figure 6: PCB Schematic

Figure 7: Bluetooth Module Pipeline

a BLE server that exposes a custom service with a writable characteristic. Upon initialization, the system configures a primary service and registers a characteristic that allows both read and write operations, serving as the main channel for remote commands.

The BLE interface is interrupt-driven. A subclass of `BLECharacteristicCallbacks` is defined to override the `onWrite()` function. Whenever the host sends a new command, the ESP32 immediately triggers this callback, which parses the received string (e.g., "+500" or "-800") and converts it into control actions for the servo motor, including rotation direction and duration.

To manage connection states, the ESP32 also registers a server callback to detect connect and disconnect events. When disconnection is detected, advertising is restarted automatically. In addition, a periodic "ping" string is sent to the host every 10 seconds using BLE notifications. This mechanism allows the host to verify that the connection remains active and facilitates lightweight keep-alive communication.

On the host side, a Bluetooth client (e.g., smartphone using *nRF Connect*) connects to the ESP32 device, discovers the BLE service, and writes UTF-8 encoded command strings to the characteristic. Commands such as "+500", "-400", or "0" represent forward, reverse, or stop signals for the servo motor respectively. The host can also receive periodic "ping" notifications, which provide feedback on connection health and device status.

Servo We selected the 360° version of the SG90 micro servo as the actuation component for the robot's jumping mechanism. This decision was motivated by multiple engineering trade-offs. The SG90 servo offers a better power-to-weight ratio than stepper motors in our application where weight is critical for the lightweight jumping robot design, even though they provide very good angular control, the stepper motor is way too heavy. There was also consideration given to coreless DC motors, but they did not provide enough torque to reliably actuate the spring-loaded mechanical structure.

The 360° SG90 is a continuous rotation servo, which processes standard PWM signals as commands for speed and direction - not as commands for target angles. A PWM signal

of 1.5ms pulse width results in zero speed (stop) and closer to 0.5ms pulse width maximum speed in one direction, and 2.5ms pulse width maximum speed in the opposite direction.

The ESP32-C3 microcontroller generates PWM actuators with a 50Hz signal using its built-in LEDC (LED Controller) Peripheral. The duty cycle is modified in order to vary the pulse width. The pulse width can vary from a 0.5ms width to a 2.5ms width. The function rotate(speedPercentMaps) takes the appropriate input speed percentage (from -100% to +100%) mapping to the appropriate pulse width. In order to rotate the servo the necessary amount of time, the BLE handler will read the incoming commands (e.g. ”+500” or ”-800”) as direction and time. The sign signifies direction, and the magnitude signifies time in milliseconds.

This straightforward but functional control interface allows responsive and accurate actuation, while also reducing hardware complexity and software complexity. The servo is powered from the output of the UDM81256 boost converter, which provides a steady 5V output under dynamic load.

2.2.3 Motion Subsystem

Driven by an SG90 micro-servo, the motion subsystem winds the twin torsional springs to a preset angle before a precision half-gear latch triggers their release. The stored energy then snaps a four-bar linkage, generating an almost-vertical launch trajectory.

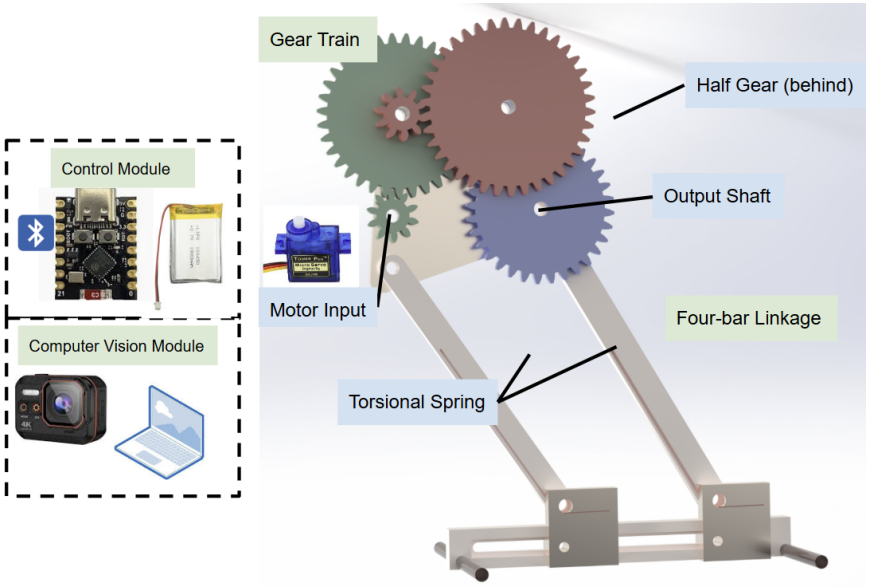


Figure 8: Physical design of the jumping robot.

Gear Transmission and Linkage Actuation To enable the jumping motion, we implemented a four-bar linkage system that transforms the motor’s rotational motion into a vertical thrust. The main purpose of using the four-bar linkage was to achieve a feasible

and controllable jumping trajectory without relying on complex mechanisms. We used simulation tools to model the linkage motion and evaluate the resulting trajectory, ensuring that the output motion remained predominantly vertical. By adjusting the lengths and pivot positions, we were able to control the initial launch angle, which in turn affected the jumping height. This tunability allowed us to explore different configurations and select a geometry that met both the size constraints and the desired performance.

The input to the linkage is driven by a motor connected to a compound gear train with a total reduction ratio of approximately $3.8 \times 3.8 \times 2 = 28.9$. This multi-stage reduction significantly lowers the motor's output speed while increasing the available torque, allowing sufficient force to preload the torsional spring before the jump. All linkage and gear components were 3D-printed and manually assembled.

Torsional Spring Selection The jumping mechanism relies on storing energy in torsional springs and releasing it rapidly. To estimate the minimum required energy for a successful jump, we apply the conservation of energy principle. The potential energy needed to lift a robot of mass $m = 30\ \mathrm{g} = 0.03\ \mathrm{kg}$ to a height of $h = 0.1\ \mathrm{m}$ is:

$$E = mgh = 0.03 \times 9.81 \times 0.1 = 0.0294\ \mathrm{J}$$

This energy must be provided by the torsional springs, which is given by:

$$E = \frac{1}{2}k\theta^2$$

where k is the torsional stiffness (in $\mathrm{Nm/rad}$) and θ is the preload angle (in radians). Since we used two identical springs in parallel, the total energy becomes:

$$E_{\text{total}} = 2 \times \frac{1}{2}k\theta^2 = k\theta^2$$

Assuming a preload angle of $\theta = \frac{\pi}{2}\ \mathrm{rad}$, the minimum spring stiffness becomes:

$$k \geq \frac{0.0294}{(\pi/2)^2} \approx 0.0119\ \mathrm{Nm/rad}$$

In our prototype, we used two torsional springs made of spring steel, each with an outer diameter of 5.5 mm and a wire diameter of 1 mm. The mean coil diameter is approximately 4.5 mm. Each spring has 6 active turns. Using the standard torsional spring stiffness formula [7]:

$$k = \frac{d^4 G}{10.8 D N}$$

where $d = 1\ \mathrm{mm}$, $D = 5.5\ \mathrm{mm}$, $G = 79.3 \times 10^9\ \mathrm{Pa}$, and $N = 6$, we estimate:

$$k \approx \frac{(0.001)^4 \times 79.3 \times 10^9}{10.8 \times 0.0055 \times 6} \approx 0.2225 \text{ Nm/rad}$$

Each spring contributes approximately 0.2225 Nm/rad, giving a combined effective stiffness of about 0.45 Nm/rad, which is sufficient for the 10cm jump.

Nonetheless, our physical testing showed that this spring configuration successfully launched a 30g robot to a height of approximately 10cm. This suggests that the theoretical estimation is conservative, and that real-world factors—such as rapid energy release, energy losses, and mechanical amplification—can compensate for lower spring stiffness.

The current design demonstrates that even relatively soft torsional springs can achieve functional jumping performance under appropriate mechanical conditions.

Motor Transmission and Integration To determine whether the selected motor is capable of compressing the torsional springs to the desired preload angle, we estimated the minimum required torque based on experimental measurements. Specifically, we measured the tangential force required to twist a single torsional spring by 90°.

The total torque needed at the output shaft (i.e., at the spring interface) is given by:

$$\tau_{\text{spring}} = F \cdot r \cdot N$$

where $F = 6\,\text{N}$ is the measured tangential force, $r = 0.04\,\text{m}$ is the moment arm, $N = 2$ is the number of torsional springs in parallel. Then we substitute the values:

$$\tau_{\text{spring}} = 6\,\text{N} \cdot 0.04\,\text{m} \cdot 2 = 0.48\,\text{N m}$$

Note that this torque will be delivered after the gear train. To calculate the corresponding required motor torque, we divide by the gear reduction ratio $G = 28.88$ and account for transmission inefficiency using a safety factor of $\frac{1}{0.8} = 1.25$:

$$\tau_{\text{motor}} = \frac{\tau_{\text{spring}}}{G \cdot \eta} = \frac{0.48}{28.88 \times 0.8} \approx 0.0208\,\text{N m}$$

Therefore, the motor must supply at least 0.0208 Nm of torque at its shaft to overcome mechanical losses and preload the springs to the desired angle. This value serves as the baseline for selecting an appropriate actuator and ensuring reliable operation under real-world conditions.

The servo under consideration weighs only 9 grams and provides a stall torque of 1.6 kg·cm (approximately 0.157 Nm). Its built-in position feedback and angle control make it well suited for use with the partial gear design.

3 Verification

3.1 Spring and Detachment Module

To enable jumping, the torsional spring must be capable of storing and releasing sufficient torque to actuate the four-bar linkage. Additionally, a passive detachment mechanism ensures the spring disengages cleanly at the correct moment, enabling rapid energy release.

Table 1: Spring and Detachment Module Requirements, Verification, and Quantitative Results

Requirement	**Verification**	**Quantitative Results**
The torsional springs must output sufficient torque ($\geq 0.075\,\mathrm{N\,m}$) when twisted from $120°$ to $30°$ to actuate the linkage mechanism for jumping.	A. The torsional spring was mounted to a shaft with a known moment arm ($0.04\,\mathrm{m}$). B. The spring was rotated to $90°$, and weights were added until equilibrium to measure the applied force. C. The spring was then released to observe its actuation of the four-bar linkage.	Measured force: $6\,\mathrm{N}$. Resulting torque: $T = F \cdot r = 6\,\mathrm{N} \cdot 0.04\,\mathrm{m} = 0.24\,\mathrm{N\,m}$. The spring successfully actuated the four-bar linkage through its full stroke upon release. **Result: Requirement met.**
The detachment mechanism must disengage automatically at full compression to release stored energy instantly.	A. A custom half-gear was used as the passive release mechanism. B. The engagement and disengagement of the half-gear mechanism were tested across the range of motion.	Optimal tooth coverage was found to be $\frac{5}{16}$ of the full gear circumference. The gear profile keeps the gear engaged from $30°$ to $120°$ and ensures disengagement occurs precisely at the maximum preload angle. Experimental trials showed consistent detachment at the desired angular position with no failure or delay. **Result: Requirement met.**

3.2 Gear Module

The gear train connects the motor to the leg of the robot to compress the torsional spring. It amplifies the motor torque to the required level for spring actuation. It must ensure

torque transmission efficiency and smooth operation without mechanical issues such as jamming or excessive backlash.

Table 2: Gear Module Requirements, Verification, and Quantitative Results

Requirement	**Verification**	**Quantitative Results**
The gear train must provide sufficient torque amplification for the motor to preload the torsional spring. The designed gear ratio is 28.88:1.	A. The gear train was assembled with the designed total reduction ratio of 28.88:1. B. The motor was actuated to rotate the spring from 30° to 120°, and observe its performance.	The motor successfully rotated the spring from 30° to 120°, confirming sufficient torque transfer. No additional gearing was required, and no motor stall was observed. The current configuration is adequate for spring loading. **Result: Requirement met.**
The gear train must mesh smoothly without jamming, abnormal noise, or significant backlash.	A. Observed the gear motion visually and acoustically during multiple continuous actuation cycles.	No instances of gear jamming or abnormal gear noise were detected. Backlash was minimal and did not negatively affect system performance. The system completed at least 5 preload and release cycles with consistent behavior. **Result: Requirement met.**

3.3 Embedded Control Module

Table 3: Control Subsystem Requirements, Verification and Quantitative Results

Requirement	**Verification**	**Quantitative Results**
The system must reliably generate PWM signals to control SG90, with direction and duration specified via Bluetooth commands.	A. Send rotation commands with specified direction and duration over BLE. B. Observe servo behavior and timing in response to each command. C. Measure actual rotation duration using stopwatch to assess consistency.	The servo consistently rotated in the correct direction for each received command, and the observed rotation durations matched the intended values within a $\pm$500 ms tolerance across 5 repeated trials in both directions. **Result: Requirement met.**
The system must establish and maintain a stable BLE connection with the host device.	A. Power on the ESP32-C3 and wait for BLE advertising to start. B. Connect to the device from a smartphone and observe connection stability over time.	The BLE module initialized successfully within 2 s after power-up and was identified as `Jumping_Robot` with a transmit power of 9 dBm and signal strength of –58 dBm, and the connection with the host smartphone was maintained continuously for over 5 minutes in a static environment. **Result: Requirement met.**
The 3.7V LiPo battery must provide sufficient voltage and current to support the operation of the ESP32-C3 and SG90 servo motor, and maintain system stability under peak load conditions.	A. Power the system using a freshly charged 401119 LiPo battery. B. Monitor servo operation and Bluetooth signal connection behavior during activation. C. Measuring the initial voltage and edge voltage of the system stable operation	The measured open-circuit voltage of the freshly charged battery was 3.9 V. Throughout repeated actuation cycles, the system remained fully functional, with no observed signal dropouts. Empirically, the system operated stably as long as the voltage remained above 3.3 V. **Result: Requirement met.**

3.4 Computer Vision Module

Table 4: Vision Subsystem Requirements, Verification and Quantitative Results

Requirement	**Verification**	**Quantitative Results**
The UHD camera must capture video frames at 30 Hz with a resolution of 3840×2160 to ensure sufficient visual detail.	A. System camera connection. B. Frame collection with OpenCV at 30 fps and 3840x2160 resolution.	The system connected the camera by OpenCV and USB. The system was able to collect consecutive frames using OpenCV at 30 fps, 3840x2160 resolution each frame. **Result: Requirement met.**
Camera intrinsic parameters must be precisely calibrated for accurate 3D reconstruction.	A. Use of OpenCV calibration functions to compute reprojection error. B. Assessment of reprojection error.	The system used calibration functions from OpenCV to compute reprojection error successfully. The reprojection error was under 3. **Result: Requirement met.**
ArUco markers at a size of 3cm×3cm must be detected up to a distance of 3 meters with a false positive rate less than 0.1%.	A. Evaluation of false positive rate.	The false positive rate was 0.057%. **Result: Requirement met.**
Vision processing must maintain latency below 100 ms per frame.	A. Measurement of end-to-end latency.	The end-to-end latency was 87ms. **Result: Requirement met.**
The system must successfully calculate the length a with an error in 10%.	A. Comparison of robot's landing and highest location with input height and distance for three times.	It was proved by comparing the landing or highest location of the robot with the input height or distance successfully for three times. **Result: Requirement met.**

4 Costs and Schedule

Cost

The calculation of Labor Costs use rates based on UIUC ECE graduate salaries ($85k/year = $41/hr, 2080 hours per year). We include 25% buffer for iterations. The information is provided in Table 5.

Table 5: Labor Cost Calculation

Role	**Hours**	**Rate ($/hr)**	**Total**
Mechanical (2 persons)	100	41	4,100
Computer Vision	50	41	2,050
Control Systems	55	41	2,255
		Total Labor	**8,405**

All materials are bought from Taobao/Tmall. We convert the units from RMB to USD. 3D Printing materials and equipments are provided by the university for free. The information is provided in Table 6.

Table 6: Parts & Materials Cost

Component	**Manufacturer**	**Description**	**Qty**	**Unit Price ($)**	**Total ($)**
ESP32C3-Supermini	Espressif	WiFi/BLE Development Board	2	2.15	4.30
SG90 Servo Motor	-	360° Mini Servo (9g)	2	0.82	1.64
401119-100 LiPo Battery	-	3.7V Mini Battery	2	0.66	1.32
Torsional Spring	Misumi	Torsion Spring (1.0 mm wire)	10	0.30	3.00
4K Camera	Hengqiaotong	USB 4K Webcam with Autofocus	1	32.44	32.44
				Total Parts	**42.70**

The grand total cost is summarized as $ 8447.70 in total.

Project Schedule

The complete timeline is detailed in Table 8.

Team Responsibilities:

- *Xuecheng Liu and Xinyi Yang*: Mechanical Systems
- *Siying Yu*: Control System
- *Hanjun Luo*: Computer Vision System

Table 7: Weekly Project Schedule (February–May 2025)

Week	Mechanical	Control	Vision	General
February 2025				
Feb 3–9	Finalize CAD, order key parts	Route PCB, outline firmware skeleton	Set camera spec	Kick-off; freeze requirements
Feb 10–16	Print & assemble first proto; basic fit	Submit PCB; PWM + BLE demo	Calibrate camera	Weekly sync
Feb 17–23	Test linkage	Bring-up PCB; BLE parsing	Prototype marker detection	Update docs
Feb 24–Mar 2	Reinforce release; metal shafts ordered	Power-path test	Depth calc mock-up	Mid-month review
March 2025				
Mar 3–9	Iterate half-gear; spring tests	Closed-loop servo ctrl	Jump-logic draft	Interface freeze
Mar 10–16	Carbon-fiber base; robustness check	Tune servo timing; safety watchdog	Param-tuning loop	Doc update
Mar 17–23	First jump trials; height metrics	Battery stress; firmware profiling	Speed optimise	Integration plan
Mar 24–30	Strength tweaks post-tests	Ready firmware for merge	Clean data output	Start subsystem merge
April 2025				
Mar 31–Apr 6	Support full-stack dry-run	BLE+vision handshake	Pipe vision → ctrl	Full link test
Apr 7–13	Monitor wear; swap weak gears	Debug integrated loop	Verify accuracy	Integrated jump demo

Table 7 continued

Week	Mechanical	Control	Vision	General
Apr 14–20	Fine-tune preload vs. height	Adjust timings + thresholds	Robustness vs. lighting	System tuning
Apr 21–27	Mechanical verification scripts	Control verification scripts	Vision verification scripts	Mid-April review
May 2025				
Apr 28–May 4	Stress-test springs; doc photos	Module verification complete	Latency benchmark	Compile results
May 5–11	Assist full-system V&V	Assist full-system V&V	Assist full-system V&V	Draft final report
May 12–18	Review mech sections	Review control sections	Review vision sections	Finalize figures tables
May 19–25	Final mech check	Final ctrl check	Final vision check	Final report + rehearsal
May 26–29	**Submit final report and give demonstration**			

5 Conclusions

5.1 Accomplishments

This project addressed key challenges of robotic locomotion in complex terrains by designing and conceptually validating a flea-inspired jumping robot that is low-cost, reliable, and highly agile. The robot meets all high-level requirements and achieves three discrete jump heights—see Figs. 9, 10, and 11 for details.

It demonstrates multi-level jumping, real-time 3-D distance estimation, and automatic link-length adjustment. Our system establishes a practical foundation for further research on multi-jump stability, spring selection optimization, and integrated compliant mechanisms.

5.2 Uncertainties and FutureWork

Although the system design has the required functions, there are still some uncertainties. First, the 3-D-printed plastic cannot tolerate the large torque generated during each launch, so the prototype usually breaks after about ten jumps. Secondly, because the current design is not enough to support the weight of the two servos, the jumping height must be achieved by manually adjusting the bottom link, which is not fully automated. Future work includes replacing PLA with aluminum to improve design durability.

While the current implementation remains a basic prototype with limited jump height and frequency, the design demonstrates the viability of performing mechanical jumps with minimal electronic control. In the future, this type of robot has potential applications in areas such as urban obstacle traversal, search-and-rescue, or even low-gravity mobility exploration.

5.3 Ethical Considerations

For ethical consideration, our camera system continues to comply strictly with privacy-preserving protocols and all testing takes place in a controlled environment with clear notifications. Through these methods, we keep our robot safe and ethical, following IEEE Code: paramount importance must be given to the safety, health, and welfare of the public.

The wider impact of this project and similar progress in agile robotics will be multifaceted. On a global scale, this type of technology can improve capabilities for disaster response, remote inspection of infrastructure, and environmental monitoring in areas never considered before. This is valuable regarding saving lives and resources. Also, there is the potential for economic impacts through innovation in specialized robotics types, which can create new markets and application spaces. We feel that what we have done here can stimulate ideas for future work.

References

[1] M. Kovac, M. Fuchs, A. Guignard, J.-C. Zufferey, and D. Floreano, "A miniature 7g jumping robot," in *2008 IEEE international conference on robotics and automation*. IEEE, 2008, pp. 373–378.

[2] C. Zhang, W. Zou, L. Ma, and Z. Wang, "Biologically inspired jumping robots: A comprehensive review," *Robotics and Autonomous Systems*, vol. 124, p. 103362, 2020.

[3] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[4] M. Aharchi and M. Ait Kbir, "A review on 3d reconstruction techniques from 2d images," in *Innovations in Smart Cities Applications Edition 3: The Proceedings of the 4th International Conference on Smart City Applications 4*. Springer, 2020, pp. 510–522.

[5] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, no. 2, 2000.

[6] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[7] J. J. Uicker, J. J. Uicker Jr, G. R. Pennock, and J. E. Shigley, *Theory of machines and mechanisms*. Cambridge University Press, 2023.

Appendix A Demonstration of Multiple Jump Heights

In our tests, jump height is defined as the vertical distance from the robot's lowest point to the ground at the apex of its trajectory.

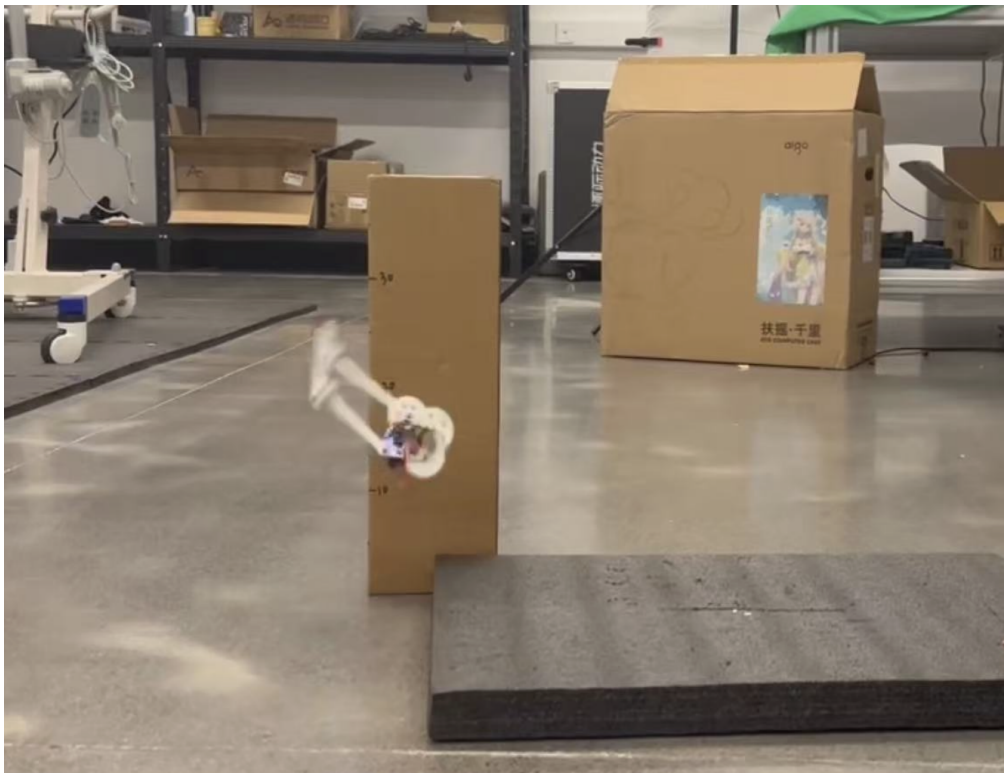


Figure 9: Jump height ≈15 cm.

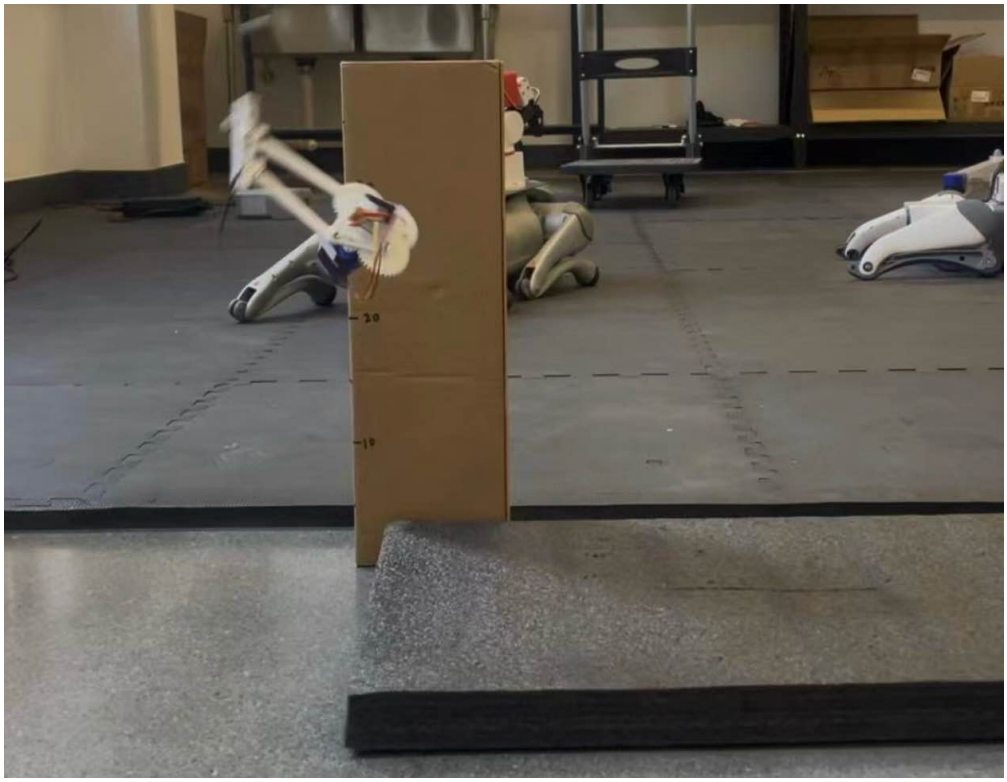


Figure 10: Jump height ≈25 cm.

Figure 11: Jump height ≈35 cm.

Appendix B Detailed Schedule

Table 8: Weekly Project Schedule (February - May 2025)

Week	Mechanical Tasks (XL, XY)	Control Tasks (SY)	Vision Tasks (HL)	All Members / General Tasks
+ 3**February 2025: Design Finalization, Component Procurement, Initial Prototyping**				
February 2025: Design Finalization, Component Procurement, Initial Prototyping				
Feb 3-9	Finalize CAD for four-bar linkage, gears (half-gear), initial base. Source materials (springs, shafts, 3D print).	Finalize ESP32-C3 circuit, PCB layout. Source ESP32-C3, SG90 servo, LiPo, boost converter, PCB components.	Finalize Aruco marker strategy (e.g., DICT_4X4_250). Select UHD camera. Plan camera calibration.	Project kickoff. Review requirements. Finalize task breakdown.
Feb 10-16	3D print initial mechanical parts. Order springs, initial shafts (e.g., plastic for trial).	Submit PCB for fabrication. Start ESP32 firmware: basic servo PWM, initial BLE setup.	Procure UHD camera. Setup OpenCV. Implement image capture. Start collecting calibration images.	Weekly progress meeting.
Feb 17-23	Assemble first mechanical prototype. Test linkage motion. Test initial release mechanism concept (pre-half-gear iteration).	Firmware: BLE command parsing. Test servo control precision.	Complete camera calibration (get K, D matrices). Implement Aruco marker detection.	Update documentation.

Continued on next page

Table 8 continued from previous page

Week	Mechanical Tasks (XL, XY)	Control Tasks (SY)	Vision Tasks (HL)	All Members / General Tasks
Feb 24 - Mar 2	Identify issues: release instability, plastic shaft weakness, base weight. Redesign: refine half-gear, order metal shafts, design carbon fiber base.	Receive & assemble custom PCB. Test PCB with ESP32, servo. Test power system (LiPo, boost converter).	Test Aruco detection robustness (distance up to 3m, lighting). Implement 3D distance calculation.	Mid-Feb review. Problem-solving session.
March 2025: Iterative Development & Subsystem Maturation				
Mar 3-9	Fabricate & test iterated release (half-gear). Test metal shafts with spring compression.	Refine BLE: interrupt handling, connection state, ping mechanism. Test basic commands.	Develop jump parameter logic based on distance (target height/distance scenarios). Use kinematic model $f_{linkage}(a, l_1, l_2, l_3)$.	Weekly progress meeting.
Mar 10-16	Assemble & test improved mechanics (half-gear, metal shafts). Fabricate & test carbon fiber base.	Implement precise servo rotation logic (map commands to PWM for speed/direction of 360° servo). Test preload/release sequence.	Refine jump parameter calc (iterative 'a' determination). Test with robot parameters (E_k, m).	Update documentation.
Mar 17-23	Initial jump tests with refined mechanics. Measure height, observe stability.	Test power management (boost converter under load, battery life est.).	Optimize vision processing for speed (¡100ms), potentially using CUDA.	Integration planning meeting. Define module interfaces.

Continued on next page

Table 8 continued from previous page

Week	Mechanical Tasks (XL, XY)	Control Tasks (SY)	Vision Tasks (HL)	All Members / General Tasks
Mar 24-30	Further mechanical refinements based on jump tests. Ensure robustness.	Prepare control system for integration. Ensure reliable command execution for spring actuation.	Prepare vision system output (e.g., length 'a' or servo commands) for control system.	Begin basic integration: Vision $\rightarrow$ Control $\rightarrow$ Mechanical.
April 2025: System Integration & Testing				
Mar 31 - Apr 6	Support integrated system testing. Make mechanical adjustments as needed.	Integrate BLE commands from vision system (or host PC relaying vision output). Test servo response to integrated commands.	Integrate vision output with control system. Test data transmission.	Full system integration. Focus on data flow and command handshaking.
Apr 7-13	Monitor mechanical components (gears, springs, linkage) during integrated tests.	Monitor servo, BLE, power during integrated operation. Debug control logic.	Verify accuracy of vision-based jump commands and resulting robot action.	Initial integrated system jump tests. Debugging.
Apr 14-20	Fine-tune mechanical aspects for jump consistency (e.g., leg length adjustments if using manual slots).	Adjust control parameters (e.g., servo timings) based on integrated tests.	Tune vision algorithms for varying conditions. Refine jump parameter calculations.	System performance tuning. Aim for multi-level jumps.
Apr 21-27	Prepare for Spring & Gear module verification based on report's Table 1 & 2.	Prepare for Embedded Control module verification based on report's Table 3.	Prepare for Computer Vision module verification based on report's Table 4.	Intensive testing & debugging. Start formal verification procedures. Mid-April review.

Continued on next page

Table 8 continued from previous page

Week	Mechanical Tasks (XL, XY)	Control Tasks (SY)	Vision Tasks (HL)	All Members / General Tasks
May 2025: Verification, Finalization, Report Writing				
Apr 28 - May 4	Complete verification for Spring and Detachment Module & Gear Module. Document mechanical design, iterations.	Complete verification for Embedded Control Module (PWM, BLE, servo torque). Document control system, PCB, firmware.	Complete verification for Computer Vision Module (camera capture, calibration, Aruco, latency, param calc). Document vision system.	Compile individual contributions for report.
May 5-11	Assist with full system verification documentation.	Assist with full system verification documentation.	Assist with full system verification documentation.	Full system verification (dimensions, jump levels, wireless trigger). Draft all report sections (Intro, Design, Verification, Costs, Conclusion).
May 12-18	Review mechanical sections of the report.	Review control sections of the report.	Review vision sections of the report.	Complete drafting of all report sections. Create/finalize figures & tables. Compile References. Write Abstract.

Continued on next page

Table 8 continued from previous page

Week	Mechanical Tasks (XL, XY)	Control Tasks (SY)	Vision Tasks (HL)	All Members / General Tasks
May 19-25	Final check of mechanical content and contributions.	Final check of control content and contributions.	Final check of vision content and contributions.	Final review of complete report. Check formatting, grammar, completeness. Prepare presentation/demo if required. TA Review/feedback.
May 26-29	**Submit Final Report**			

Appendix C Algorithms for Different Scenarios

Scenario 1: Achieving a Target Jump Height In this scenario, the objective is to reach a specific jump height h, representing the apex of the parabolic trajectory. The target height h is provided as an input.

1. **Calculate Required Launch Angle (θ_0):** Given v_0 and the target height h, the required launch angle θ_0 (measured from the horizontal) is found using the projectile motion equation for maximum height:

   $$h = \frac{v_0^2 \sin^2(\theta_0)}{2g}$$

   Solving for θ_0:

   $$\sin(\theta_0) = \frac{\sqrt{2gh}}{v_0}$$

   $$\theta_0 = \arcsin\left(\frac{\sqrt{2gh}}{v_0}\right)$$

   This calculation is valid if $v_0^2 \geq 2gh$; otherwise, the target height is unreachable with the given E_k.

2. **Determine Horizontal Range (x):** Once θ_0 is known, the corresponding horizontal range x for this jump can be calculated:

   $$x = \frac{v_0^2 \sin(2\theta_0)}{g}$$

3. **Determine Variable Link Length (a):** The calculated θ_0 is the target launch angle. The appropriate variable link length a is then determined by finding an a such that $f_{linkage}(a, l_1, l_2, l_3) \approx \theta_0$. This may involve solving the inverse kinematics for the linkage or iterating through the permissible range of a to find the value that produces the launch angle closest to θ_0.

Scenario 2: Achieving a Target Horizontal Distance (x_{target}) over Two Jumps This scenario involves executing two distinct jumps to cover a specified horizontal distance. For each jump $j \in \{1, 2\}$, a target horizontal distance x_j is provided (e.g., from the CV system identifying segments of a path). The goal is to calculate a different variable link length a_j for each jump. The initial launch velocity v_0 (derived from E_k) is assumed to be the same for both jumps.

For each jump j:

1. **Calculate Required Launch Angle ($\theta_{0,j}$):** Given v_0 and the target horizontal distance for this jump x_j, the required launch angle $\theta_{0,j}$ is found using the projectile motion equation for range:

   $$x_j = \frac{v_0^2 \sin(2\theta_{0,j})}{g}$$

Solving for $\theta_{0,j}$:

$$\sin(2\theta_{0,j}) = \frac{gx_j}{v_0^2}$$

$$\theta_{0,j} = \frac{1}{2}\arcsin\left(\frac{gx_j}{v_0^2}\right)$$

This calculation is valid if $v_0^2 \geq gx_j$. For a given $x_j < v_0^2/g$ (maximum range), two solutions for $\theta_{0,j}$ exist (a low and a high trajectory). Typically, one is chosen based on criteria like obstacle clearance or energy efficiency (often the lower angle, $\theta_{0,j} < 45°$).

2. **Determine Corresponding Jump Height (h_j):** The peak height h_j for this jump trajectory is:

$$h_j = \frac{v_0^2 \sin^2(\theta_{0,j})}{2g}$$

3. **Iterative Determination of Variable Link Length (a_j):** To find the optimal variable link length a_j that produces the target launch angle $\theta_{0,j}$, an iterative approach is employed. The length a is varied within its allowed operational range (e.g., from a_{min} to a_{max}) in discrete steps (e.g., $\Delta a = 0.001\,\text{m}$ or 0.1 cm). For each candidate length a_k in this iteration:

   - The resulting launch angle $\theta_{launch}(a_k) = f_{linkage}(a_k, l_1, l_2, l_3)$ is calculated using the linkage's kinematic model.
   - The difference $|\theta_{launch}(a_k) - \theta_{0,j}|$ is evaluated.

   The value a_k that minimizes this difference (i.e., produces a launch angle closest to the required $\theta_{0,j}$) is selected as the optimal a_j for that jump. This process is repeated to find a_1 and a_2 for the two jumps.

This systematic calculation ensures that the robot's jumps are tailored to the specific requirements of the terrain and obstacles identified by the vision system.